Gates & Cooper LLP

Howard Hughes Center
6701 Center Drive West, Suite 1050
Los Angeles, California 90045

RECEIVED
CENTRAL FAX CENTER
OCT 2 6 2004

FAX TRANSMISSION TO USPTO

TO: Commissioner for Patents
Attn: Examiner Keshaba C. Sahoo
Patent Examining Corps
Facsimile Center
Alexandria, VA 22313-1450

FROM:	Jason S. Feldmar
OUR REF.:	G&C 30566.90-US-01
TELEPHONE:	(310) 642-4141

Total pages, including cover letter: 26

PTO FAX NUMBER: (703) 872-9306

If you do NOT receive all of the pages, please telephone us at (310) 641-8797, or fax us at (310) 641-8798.

Title of Document Transmitted:	**BRIEF OF APPELLANT AND AUTHORIZATION TO CHARGE DEPOSIT ACCOUNT IN THE AMOUNT OF $340.00 FOR FILING FEE.**
Applicant:	Keshaba C. Sahoo
Serial No.:	09/560,434
Filed:	April 27, 2000
Group Art Unit:	2177
Title:	INTELLIGENT OBJECT VERSIONING
Our Ref. No.:	G&C 30566.90-US-01

Please charge all fees to Deposit Account No. 50-0494 of Gates & Cooper LLP.

By: [illegible]
Name: Jason S. Feldmar
Reg. No.: 39,187

I hereby certify that this paper is being transmitted by facsimile to the U.S. Patent and Trademark Office on the date shown below.

[illegible]
Signature

10/26/04
Date

BEST AVAILABLE COPY

JSF/sjm

G&C 30566.90-US-01

Confirmation No.: 4313
Due Date: October 26, 2004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

RECEIVED
CENTRAL FAX CENTER
OCT 2 6 2004

Applicant:	Keshaba C. Sahoo	Examiner:	Mirande Le
Serial No.:	09/560,434	Group Art Unit:	2177
Filed:	April 27, 2000	Docket:	G&C 30566.90-US-01
Title:	INTELLIGENT OBJECT VERSIONING		

CERTIFICATE OF MAILING OR TRANSMISSION UNDER 37 CFR 1.8

I hereby certify that this correspondence is being facsimile transmitted to the United States Patent and Trademark Office, Fax No. (703) 872-9306 on October 26, 2004.

Signature: [illegible]
Typed Name of Person Mailing this Certificate: Suzie McCleave

MAIL STOP APPEAL BRIEF – PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

We are transmitting herewith the attached:

☒ Transmittal sheet, in duplicate, containing a Certificate of Mailing or Transmission under 37 CFR 1.8.
☒ Brief of Appellant.
☒ Charge the Fee for the Brief of Appellant in the amount of $340.00 to the Deposit Account.

Please consider this a PETITION FOR EXTENSION OF TIME for a sufficient number of months to enter these papers, if appropriate.

Please charge all fees to Deposit Account No. 50-0494 of Gates & Cooper LLP. A duplicate of this paper is enclosed.

Customer Number 22462
GATES & COOPER LLP
Howard Hughes Center
6701 Center Drive West, Suite 1050
Los Angeles, CA 90045
(310) 641-8797

By: [illegible]
Name: Jason S. Feldmar
Reg. No.: 39,187
JSF/sjm

Due Date: October 26, 2004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:)
Inventor: Keshaba C. Sahoo) Examiner: Miranda Le
Serial #: 09/560,434) Group Art Unit: 2177
Filed: April 27, 2000) Appeal No.: ________
Title: INTELLIGENT OBJECT VERSIONING)

RECEIVED
CENTRAL FAX CENTER
OCT 2 6 2004

BRIEF OF APPELLANT

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In accordance with 37 CFR §41.37, Appellant hereby submits the Appellant's Brief on Appeal from the final rejection in the above-identified application, as set forth in the Office Action dated May 27, 2004.

Please charge the amount of $340 to cover the required fee for filing this Appeal Brief as set forth under 37 CFR §41.37(a)(2) and 37 CFR §41.20(b)(2) Deposit Account No. 50-0494 of Gates & Cooper LLP, the assignee of the present application. Also, please charge any additional fees or credit any overpayments to Deposit Account No. 50-0494.

I. REAL PARTY IN INTEREST

The real party in interest is Autodesk, Inc. the assignee of the present application.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences for the above-referenced patent application.

III. STATUS OF CLAIMS

Claims 1-3, 5-7, and 9-11 are pending in the application.

Claims 1-3, 5-7, and 9-11 were rejected under 35 U.S.C. §102(e) as being anticipated by Saether et al., U.S. Publication No. US 2001/0042073 A1 (Saether).

All of these rejections are being appealed.

IV. STATUS OF AMENDMENTS

No amendments to the claims have been made subsequent to the final Office Action.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Applications (such as word processing programs, spread sheet programs, or drawing programs) are used to create files (e.g., a documents, spreadsheets, or drawings) (see page 2, lines 15-21). The created files often contain objects (e.g., a drawing file may contain a line object, shape object, circle object, arc object, etc.) (see page 2, lines 19-21). When saving a file, the data for the objects within the file are streamed out with the file (see page 2, lines 21-22). In addition, the file is related to the version of the application that created the file. However, objects may also evolve and have different versions over time. In this regard, if the user desires to store a file in an earlier version than the object was first introduced, the evolution of the object may be hindered (see page 3, lines 1-7).

Independent claims 1, 5, and 9 are generally directed to storing object data in a particular version (see page 4, lines 8-12). The claims provide that the representative data (i.e., for the object) comprises actual methods and attributes of the object (see page 7, lines 20-21). The claims further provide for storing a particular version of the object by streaming data for a particular version of the object out to a file (see page 4, lines 8-12).

More specifically, the version of the file (that the object is being stored in) is compared to the version of the object when the object was introduced/originated (step 500 of FIG. 5, and page 10, lines 10-14). If the file version is the same or newer than when the object was first introduced, the file is saved such that (the data representing) the instance of the object is streamed out (e.g., stored) in the file version (step 506 and page 10, lines 15-20).

However, if the file version is older than when the object was introduced, the file is saved such that the (data representing) the instance of the object is streamed out (e.g., stored) in the object introduction version (see steps 502 and 504, page 10, line 21-page11, line 5).

Dependent claims 2, 6, and 10 further provide that if the file version is earlier than the object introduction version, the object is represented as a proxy object (when a file is opened), and when the file is saved, the proxy object holds onto the object's data and streams out the object's data (see steps 502 and 504 and page 11, lines 6-19).

Dependent claims 3, 7, and 11 provide the ability for superior objects to query the object to determine the appropriate version to stream out to the file (see step 306 and page 9, lines 15-21). The object responds to the superior object (that actually performs the streaming out operation) to stream out in the requested file version (if the requested file version is equal to or later than the object introduction version) and in the object introduction version (if the requested file version is earlier than the object introduction version) (see page 9, line 22-page 10, line 1; page 10, line 17-page 11, line 5).

VI. GROUNDS OF REJCTION TO BE REVIEWED ON APPEAL

Claims 1-3, 5-7, and 9-11 stand rejected under 35 U.S.C. § 102(e) as being anticipated by Saether et al., U.S. Publication No. US 2001/0042073 A1.

VII. ARGUMENT

A. Independent Claims 1, 5, and 9 Are Patentable Over Saether

Independent claims 1, 5, and 9 were rejected as follows:

> As to claims 1, 5, and 9, Saether teaches "obtaining a request to save a file in a requested file version, wherein the file contains an object" at [0008,0009], [0031-0032], [0042-0043], [0074, 0075], [0084-0085];
>
> "determining if the requested file version is earlier than an object introduction version of the object" at [0032-0033], [0035], [0044], [0049-0054], [0064], [0077, 0078];
>
> "saving the file by streaming out data representing an instance of the object to the file in the requested file version if the requested file version is equal to or later than the object introduction version, wherein the data comprises actual methods and attributes of the object" at [0013], [0031-0033], [0034-0035], [0055, 0056], [0058], [0064, 0065], [0078];

"saving the file by streaming out the data representing the instance of the object to the file, in the object introduction version if the requested file version is earlier than the object introduction version" at [0031-0035], [0056], [0061], [0064], [0078].

Appellant traverses the above rejection for one or more of the following reasons:

(1) Saether does not teach, disclose or suggest a file containing an object as claimed;

(2) Saether does not teach, disclose or suggest determining if a requested file version is earlier than an object introduction version of an object in a file;

(3) Saether does not teach, disclose or suggest saving a file by streaming out data representing an instance of an object;

(4) Saether does not teach, disclose or suggest streaming out such data that includes/comprises actual methods and attributes of the object;

(5) Saether does not teach, disclose or suggest saving a file by streaming out data representing the instance of the object in a requested file version if the requested file version is equal to or later than the object introduction version; and

(6) Saether does not teach, disclose or suggest saving a file by streaming out data representing the instance of the object in the object introduction version if the requested file version is earlier than the object introduction version.

Independent claims 1, 5, and 9 are generally directed to storing object data in a particular version. Specifically, a file often contains objects. When the file is saved, the object (i.e., data representing an instance of the object) is streamed out/saved as part of the file. The claims also provide that the representative data comprises actual methods and attributes of the object. The claims further provide for storing a particular version of the object by streaming data for a particular version of the object out to a file.

More specifically, the version of the file (that the object is being stored in) is compared to the version of the object when the object was introduced/originated. If the file version is the same or newer than when the object was first introduced, the file is saved such that (the data representing) the instance of the object is streamed out (e.g., stored) in the file version. However, if the file version is older than when the object was introduced, the file is saved such that the (data representing) the instance of the object is streamed out (e.g., stored) in the object introduction version. The cited reference does not teach nor suggest these various elements of Appellant's independent claims.

The cited reference does not teach or suggest these various elements of Appellant's independent claims.

Saether merely describes a method and system for managing the replication and version synchronization of updates to a set of source files on geographically distributed heterogeneous content servers with minimal impact on a network's bandwidth. The configuration of each content server is either manually entered or automatically determined. The current version of the source files are created on at least one source server. A Primary global server stores a copy of the current version of the set of the source files along with the configuration of each content server. The Primary global server generates and distributes a particular version change container and version distribution list to each remotely located Secondary global server. Each Secondary global server employs the version distribution list and the contents of the version change container to identify the current version of each source file necessary to upgrade the set of source files on each local content server. Each identified source file is copied to a sub-directory on each local content server associated with the Secondary global server. At each local content server, the renaming of each copied source file is employed to update to the current version of the set of source files on the content server. A versioned file tree repository for the set of source files includes archived objects. When the version distribution list identifies a previous version, the current version of source files on the local content servers can be rolled back to the previous version (see Abstract).

Firstly, Saether's use of objects are not even remotely similar to that used in the presently claimed invention. Paragraph [0008] describes Saether's use of an object:

> [0008] In accordance with other aspects of the present invention, the method provides for archiving each version of the set of source files in a repository on the global server, the archiving causing each source file to be individually compressed and stored as an **archived object** in the repository associated with the global server. The repository can be a versioned file tree repository for the set of source files. (Emphasis added)

As used in Saether, a source file may be compressed and stored as an archived object in a repository. Thus, while Saether provides for storing a file as an object, the present invention provides for a file that contains an object. Further, the file is saved by streaming out methods and attributes of the object. Accordingly, Saether's object use is completely different and distinguishable from a file that contains an object as claimed.

In response to this argument (asserted in response to a prior Office Action), the final Office Action provides:

> Per (1), Saether teaches a file containing an object at [0008], [0075], [0074], [0084]. Saether teaches a source file is compressed and stored as an object in a repository associated with the global server [0008]; however, it should be understood that an archived source file is also a file, e.g., F1.RCA, F2.RCA (RCA is a file extension), ([0075]). Each file has a version value (i.e., "1.2", "1,1", [0079]) associated with the file, and the file version contain object (i.e., GIF, TIFF, AVI, JPEG, MPEG, HTML pages, Java scripts,..., application programs), ([0084]).

Appellant agrees that Saether teaches a source file is compressed and stored as an object in a repository. However, as stated above, the claims provide for (1) a file containing an object, and (2) the object has methods and attributes. Saether fails to teach such object use. The final Office Action states that "it should be understood that an archived source file is also a file". However, Appellant questions the relevance of such a statement. The claims do not provide such a limitation and Appellant has not attempted to argue that an archived source file is not a file. Appellant also agrees that each of Saether's archived files may have a version value (see paragraph [0079]). However, the final Office Action then asserts that the file version contain objects and cites paragraph [0084]. Paragraph [0084] provides:

> [0084] It is to be understood that embodiments of the present invention can be created to support all file based content and applications including GIF, TIFF, AVI, JPEG, MPEG, HTML pages, JAVA scripts, Active Server pages, postscript document format (PDF), ActiveX, JAVA, and application programs. It is envisioned that embodiments of the present invention provides security mechanisms for protecting the delivery of content and application files to content servers. These security mechanisms enable remote administration of the present invention through a secure shell command line (SSH) and a secure socket layer (SSL) for browser based administration.

However, while Saether may support particular file based content and applications, nothing in paragraph [0084] even remotely suggests saving a file that contains an object. In this regard, merely supporting file based content such as ActiveX or Java does not teach the use or saving of a file that contains an object.

The second claimed element (of claim 1) provides for determining if a requested file version is earlier than an object introduction version of an object (that is contained in a file). In rejecting this element, the Office Action relies on Saether paragraphs [0032-0033], [0035], [0044], [0049-

0054], [0064], and [0077,0078]. However, none of the cited portions even remotely describe this claim element.

In paragraph [0032], the only comparison conducted is to determine a difference between an updated version of a set of source files (stored in a versioned file tree repository on a Primary global server) and the current version of the set of source files (stored in another versioned file tree repository on each Secondary global server). Accordingly, there is no comparison conducted with respect to a file version and an introduction version of an object contained within the file. Nowhere in paragraph [0032] is there any reference, implicit or explicit to an introduction version of the object.

Paragraph [0033] merely provides for generating a current version of source files and files to be removed for each local content server based on an update version identified in a version delivery list. However, there is no determination as claimed. Further, there is no object introduction version even remotely mentioned or described implicitly or explicitly.

Paragraphs [0035] and [0064] merely describe how servers may rollback to a previous version and how source files (that didn't exist in a previous version) may be deleted after copying to a local content server. Again, there is no reference, implicit or explicit, regarding a determination if a requested file version is earlier than an object introduction version of an object within a file. In fact, these paragraphs do not describe or refer to an object or object introduction version whatsoever.

Paragraphs [0042-0043] provide for copying only those source files from source servers that are determined to be different than the set of source files stored in a versioned file tree repository on a Primary global server. Again, such a determination does not even remotely teach the determination as claimed – there is no object or object introduction version even mentioned in either paragraph.

Paragraph [0044] describes the creation of a container based on differences between the current version of a set of source files (stored in a repository) and a set of source files stored in another repository. However, again there is no determination if a file version requested by a user is earlier than an introduction version of an object contained within a file. There is no object, or object introduction version mentioned in paragraph [0044].

Paragraphs [0049-0054] merely describe a comparison between different files. Specifically the paragraphs describe comparing information for a source file to the current version of a source file. In this regard, different source files are compared to each other. Such a comparison is significantly distinguishable from comparing a requested file version (and not a current version or existing version) to an object introduction version for an object within the file.

Paragraph [0078] merely refers to FIG. 6B which illustrates a file tree 200' and a versioned file tree repository 208'. However, there is no determination made in FIG. 6B (or in paragraph [0078]) nor is there any mention, implicit or explicit of an object introduction version. Similarly, paragraph [0079] merely provides for storing every previous version of a file within an archived source file. Again, such a teaching fails to disclose any determination, and fails to describe when and if a file/object was first introduced. In this regard, paragraph [0079] (and the remainder of Saether), fails to disclose any determination with respect to the different versions of the files stored in the archive. More specifically, Saether fails to disclose any comparison that evaluates or looks at the first version of the file or any object within the archive and comparing it to a requested file version. Such a comparison is clearly and completely lacking from Saether.

In response to similar arguments, the final Office Action provides:

> Per (2), Saether teaches the version of an object (i.e., the version container, [0044]) is generated to store object in a particular version (i.e., current version). Note that the version change container includes all the actual file data, and a portion of the file data (i.e., an instance of an object) for each existing source file that was modified in the current of the set of source files (i.e., F1.RCA, F2.RCA). As discussed in (1), the file version corresponds to <F1, 1.1> ([0077]) or <F1, 1.2 ([0079]), wherein "1.1", or "1.2" are versions of the file F1 (i.e., F1.RCA). The step of determining the different version of the file and the object, which belongs to the file, is shown at [0051]. It should be noted that after indicating the two file sources are different (i.e., the current and new version), the primary global server generates a version change container (i.e., a modified portion of the file data as mentioned) that may include a reference value (i.e., version number) associated with the current version of the set of source files [0054]. This implies the file version is earlier than the object version (i.e., container version).

The assertions in the Office Action clearly fail to address all of the claim language. The claim language specifically recites "object introduction version of the object". In other words, the information used in the comparison is the version in which the object (that is contained in the file) was introduced. Under MPEP §2142 and 2143.03 "To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970)." In this regard, the prior art must teach the claim language that includes "object introduction version".

To teach these claim elements, the Office Action merely describes comparing two different source files, creating a change container that includes a version number associated with the current version of the set of source files, and then states "this implies the file version is earlier than the **object version** (i.e., container version)." (Emphasis added). As can be clearly seen, the Office Action simply omits the claimed word "introduction" from the words "object version" when evaluating the claims in view of Saether. Further, comparing two file versions to determine their differences does not even remotely describe the specific comparison between a requested file version an object introduction version as claimed.

Again, the comparison is conducted between a requested file version and an object introduction version of an object contained within the file. Saether fails to evaluate any objects within the file. Instead, Saether appears to evaluate the file versions themselves. In this regard, assuming that Saether discloses objects within files with its teaching of Java and ActiveX from paragraph [0048] (as asserted in the final Office Action)(but which Appellant traverses), to teach the claimed comparison/determination, Saether must look at the version of one of the Java objects or ActiveX objects and not look at the file version itself as asserted in the final rejection. Saether fails to conduct any such determination/comparison.

The third and fourth elements of the claims describe the process for saving a file. Both steps save a file by streaming out data representing the instance of the object to the file. Further, the data comprises actual methods and attributes of the object. However, in the third element, the requested file version of the object is streamed out, while in the fourth element, the object introduction version of the object is streamed out.

The Office Action rejects the third and fourth element relying on paragraphs [0013], [0031-0035], [0055]0056], [0058], [0061], [0064,0065], and [0078].

Paragraph [0013] merely provides for copying differences between source files located on secondary remote servers to a container in a primary global server. The container is then distributed from the primary global server to the secondary severs so they can update their files accordingly.

However, this paragraph does not even remotely describe streaming out data to a file, streaming out data that represents an instance of an object, streaming out data that comprises actual methods and attributes of the object, or streaming out data in a particular format.

Examining paragraphs [0031-0035], Appellant notes that these paragraphs completely fail to provide for streaming out data. Instead, these paragraphs merely provide for copying various source files into particular directories and servers. There is no description of a file streaming out data whatsoever.

Further, the claims specifically provide for streaming out data representing an instance of an object that is contained within a file. The cited paragraphs do not describe any such object or instance of an object. In fact, an electronic search of Saether for the term "instan" provides no results whatsoever. Without even mentioning the word instance or referring to an instance of an object, Saether cannot possibly teach, implicitly or explicitly, a claim that specifically provides for streaming out data that represents an instance of an object. As stated, above, under the MPEP and existing case law, all words in a claim must be considered in judging the patentability of that claim against the prior art. The object-oriented aspect of the claims and the use of an object and object introduction versions cannot merely be ignored. Instead, the prior art must teach every element and word including the "object" aspects of the claims. Saether fails to do so.

Similar to paragraphs [0031-0035], paragraphs [0055, 0056], [0064] and [0078] also fail to teach streaming out data and/or a data representing an instance of an object. Instead, paragraphs [0055, 0056] refer to FIG. 3B and the process of the global server archiving (compressing) a version change container, sending the container to secondary servers, and un-archiving the version change container. However, there is no description of streaming out data (representing an instance of an object and comprising actual methods and attributes of the object) in a particular version to a file.

Paragraph [0064] merely describes FIG. 4 and provides for sending a version delivery list (indicating a previous version of the set of source files stored in a versioned file tree repository on the global server) to a global server. Further paragraph [0078] (as described above) merely describes FIG. 6B that illustrates a file tree 200' and a versioned file tree repository 208'. However, such a teaching does not describe, teach, or suggest, the claimed elements for the reasons stated above. Further, these cited portions also fail to describe streaming out in a requested file version if the requested file version is equal to or later than an object introduction version AND streaming out in an object introduction version if the requested file version is earlier than an object introduction version. In fact, nowhere in Saether is there any description of such claim elements, implicitly or explicitly.

In response to the above arguments, the final Office Action provides:

> The claimed limitations "saves a file by streaming out data representing an instance of an object" corresponds to the step of unarchiving (i.e., stream the version out) the version change container (i.e., instance of an object) and copying each unarchived source file to a new version in the versioned file tree repository on each secondary global server [0056]. It should be noted that the version change container (i.e., object introduction version) is transmitted to second global server under TAR ([0055]) format, so, the second global server needs to unarchive it before updating (i.e., saving) the version of the set of source files [0061].

In view of these arguments, it appears that the Patent Office is equating a version change container with an instance of an object. However, Appellant respectfully disagrees with and traverse such a statement. The claims specifically provide for an object in an object-oriented computer system. It is known in the art that object oriented objects have both methods and attributes and are defined using classes that are abstract. When an instance of a class/object is created, values for the various attributes are defined for that particular instance (see http://www.techweb.com/encyclopedia/). The claims specifically provide for an instance of an object and that the data streamed out comprises the actual methods and attributes of the object. On the other hand, paragraph [0044] of Saether describes the version change container:

> The version change container references the names of all of the source files that are included in or deleted from the current version of the set of source files. The version change container also includes the actual file data for each new source file and a portion of the file data for each existing source file that was modified in the current version of the set of source files.

Based on this description, it is clear that the version container is not an object oriented object and does not represent an instance of an object oriented object. Instead, the container merely references the names of files and includes file data for particular source files that have been modified. Such a description does not even remotely describe object oriented objects or a particular instance of an object. In fact, Saether completely fails to address, describe, or use any concepts associated with object-oriented programming.

Further, the claims specifically provide for an object introduction version of the object-oriented object. The rejection asserts that the object introduction version is also equivalent to the version change container. In this regard, the rejection asserts that streaming out the data representing the object in the object introduction version is equivalent to Saether's unarchiving before updating the version of the set of source files (relying on paragraph [0061]). However, streaming out data representing an instance of an object that comprises actual methods and attributes of an object (as claimed) is not similar, nor suggested or obvious, in view of a teaching that merely unarchives a file in an archive. No such teaching is even remotely present in Saether.

In addition, the Office Action cites Saether paragraph [0061]. However, paragraph [0061] merely describes sending an encrypted message from a primary server so a secondary server to update a version of the set of source files stored on each local content server by renaming source files. Nowhere in such a teaching is there any indication of unarchiving that occurs.

The Office Action continues and provides:

> Per (4), Saether teaches streaming out such data that comprises actual methods and attributes of the object at [0011], [0084]. Note that as discussed in (1), a portion of the source file (i.e., object) in the change version container could be script type ([0011], [0084]), or application program code, JAVA scripts ([0011],[0084]. Therefore, the object obviously comprises actual methods and attributes of the object (i.e., functions, method and parameters).

Appellant respectfully disagrees and traverses such statements. Paragraph 11 provides:

> [0011] In accordance with other aspects of the present invention, the method provides for employing a file access protocol to gain file level access to each source file, including FTP, NFS, CIFS and MFTP. The file access protocol may employ one port to send and receive data that includes a message and a source file. The type of source file includes image, hyper text mark-up language (HTML), script, sound, video, text, picture and application program code.

Paragraph [0084] provides:

> [0084] It is to be understood that embodiments of the present invention can be created to support all file based content and applications including GIF, TIFF, AVI, JPEG, MPEG, HTML pages, JAVA scripts, Active Server pages, postscript document format (PDF), ActiveX, JAVA, and application programs. It is envisioned that embodiments of the present invention provides security mechanisms for protecting the delivery of content and application files to content servers. These security mechanisms enable remote administration of the present invention through a secure shell command line (SSH) and a secure socket layer (SSL) for browser based administration.

The Office Action suggests that a portion of the source file could be a script or application program code. However, the conclusion that "the object obviously comprises actual methods and attributes of the object" is not even remotely obvious. Firstly, if the rejection were based on obviousness, the rejection should be under 35 U.S.C. §103 and not 35 U.S.C. §102. Thus, based on the pending rejection, the question is whether the streaming out data representing an instance of an object containing actual methods and attributes of an object is anticipated by unarchiving a file containing a script type, application program code, or Java script. Such an assertion is completely without merit. In addition, as stated above, Saether does not save a file by streaming data representing an object out to the file. More specifically, Saether does not stream out actual methods and attributes of an object out to a file in a particular version. In this regard, under an anticipation or obviousness standard, Saether still fails to teach the claimed invention.

The Office Action is attempting to break up individual aspects of the claims and find different parts of Saether to read on discreet elements. Such an approach to a rejection is improper. Under MPEP 2141.01 and 2142, the invention as a whole must be examined. Viewing the claimed invention as a whole, a file exists and a request is made to save the file in a particular file version (referred to in the claims as a "requested file version"). To save the file in the particular requested version, data representing an instance of an object that is contained in the file is streamed out. The data that is streamed out comprises the actual methods and attributes of the object. Further, the version that is streamed out is dependent on a particular comparison of the requested file version to the version when the object was first introduced. This whole invention has multiple steps and is very specific.

The Office Action is attempting equate the above claim language with a program that provides for automatically updating versions of source files on multiple content servers (see paragraph [0001]) wherein the source file may be an HTML, script, sound, video, picture, and application program code by sending an archive of the differences between various versions (i.e., the version change container)(see paragraphs [0011] and [0084]). Such an equivalency cannot be established, implicitly or explicitly. There is no mention or description of instances of objects or methods and attributes of objects in Saether. The claims specifically provide that the data that is streamed out represents an instance of an object and comprises actual methods and attributes of the object. Saether's teaching that a source file may be a script, sound, video, etc. (from paragraph [0011] does not teach or suggest streaming data out as claimed. The only portion of Saether that specifically describes the various type of source files is in paragraph [0011] and it includes HTML, script, sound, video, text, picture, and application program code. None of these file types expressly include objects as claimed.

In addition, the mere statement that embodiments can be created to support applications including Java, ASP, and HTML pages (as in paragraph [0084]) does not teach or suggest anything with respect to streaming out data as claimed. It is unknown how Saether could be extended to provide such support. There is a large variety of ways in which such support could be enabled, none of which would read on the presently claimed invention. For example, the Java application or ActiveX application could merely be used to ensure security for the application (see paragraph [0084]). Further, it is unknown if such an extension would comprise streaming out data representing an instance of an object that comprises actual methods and attributes of an object in a file. Instead, such an invention could entail storing metadata for an object or sending a text based file while leaving object information out (and assuming the recipient already has knowledge of the object). To assume that the mere disclosure of a possible extension to an application program that may utilize an object anticipates a specifically claimed implementation is wholly without merit.

The Office Action continues and states:

> Per (5), Saether teaches "saving a file by streaming out data representing the instance of the object in a requested file version if the requested file version is equal to or later than the object introduction version" at [0065], [0064]. This step corresponds to "rolling back" the current version of the set of source files to a previous version. This step also includes the step of streaming the container (i.e., the instance of the object as mentioned) to a previous version. In this case, the previous file version and previous container version are equal.

The claims specifically provide for saving in the requested file version if the requested file version is equal or later than the version when the object was introduced. The "rolling back" described in Saether provides for:

Step 170 - sending a version delivery list (indicating a previous version stored on the secondary server) from the primary server to secondary servers;

Step 172 – copying the previous version of the modified source files and restoring removed source files from the previous version; and

Step 174 – renaming the previous versions and deleting any newer versions.

Such a teaching does not even remotely describe saving a file by streaming certain data out in a particular version. Instead, various files are copied and renamed. Such copying and renaming cannot and does not anticipate or render obvious streaming out data in a requested file version if a requested file version is equal or later than the object introduction version for the object within the file. In this regard, there is no comparison conducted in Saether between a requested file version and the version of an object stored within the file itself.

The Office Action then continues and provides:

> Per (6), Saether teaches "saving a file by streaming out data representing the instance of the object in the object introduction version if the requested file version is earlier than the object introduction version" at [0058], [0061], [0056]. The step of streaming out object data correspond to the step of unarchiving the version change container ([0056]) and copying each unarchived source file to a new version in the versioned file tree repository on each secondary global server. And, the step of saving new file version corresponds to the step of updating ([0061]).

As stated above, unarchiving a file is not even remotely similar to streaming out data representing an instance of an object in a particular file version as claimed. Further, the copying of each unarchived file to a new version on a secondary server also fails to teach the claimed streaming of data. Again, there is no comparison between a requested file version and an introduction version of an object contained within the file. Further, the claims provide that if the file version is earlier than the object introduction version (i.e., the object was first introduced after the requested version of the file), then the object is streamed out in the object introduction version. Saether merely describes copying an unarchived source file to a new version on a server. There is no suggestion, implicit or explicit, regarding the specific claim limitations, their benefit, or their advantages. The Office Action then continues and states that the step of saving the new file version corresponds to the step of updating. However, it is unclear what "step of updating" the Examiner is referring to. The independent claims do not recite a step of "updating".

The Office Action further continues and states that no patentable weight was given to the recitation of "object-oriented" because it is recited in the preamble. Appellant submits that such an assertion is without merit. Firstly, independent claim 5 directly recites the term "object-oriented" in the claim elements and not the preamble. Secondly, independent claim 9 recites "object-oriented" after the term "comprising" in the claim language and then sets forth the individual method steps performed by the article of manufacture thereafter. Accordingly, claim 9 also recites the term "object-oriented" outside of the preamble. With respect to independent claim 1, the context of the claims clearly indicate the use of an object in an object-oriented environment and the specification supports such an interpretation.

The Office Action then admits Saether does not specifically suggest the term "object oriented" but discloses:

> a method for updating a version of a set of source files stored on a content server over a network, wherein the set of source files based content and applications includes GIF, TIFF, AVI, JPEG, MPEG, HTML pages, JAVA, JAVA scripts...([0084]), wherein an object (i.e., JAVA scripts) corresponds to a particular version (i.e., an instance of object) of a set of files. One skilled artisan would understand that as related to C++, JAVA is based on C++ but optimized for the distribution of program objects in a network such as the Internet, is also a programming language in the object-oriented environment.

As stated above, the mere suggestion in Saether paragraph [0084] that embodiments can be created to support all file based content and application including JAVA scripts does not even remotely suggest the detailed claim limitations set forth in the independent claims. The language in Saether suggests that the JAVA scripts could be part of the application programs supported. However, as stated above, such support could merely be in the form of security as set forth in [0084].

Paragraph [0011] is the only location that expressly indicates that the source files include image, HTML, script, sound, video, etc. However, again, the fact that a version of a file and not the version that such images, HTML, script, sound, video, are used in Saether clearly teaches away from looking at the introduction version of such HTML, script, sound, video, etc. and streaming out data (representing such HTML, script, sound, video, etc.) in a particular version based on a comparison of such HTML, script, sound, video, etc. with a requested file version. No such teaching is even remotely contemplated in Saether, implicitly or explicitly. Further, while a skilled artisan would understand that JAVA is based on C++, there is no suggestion or understanding that a skilled artisan would know to stream out data representing an instance of an object in a particular version after conducting a detailed comparison as set forth in the claims.

In view of the above, Saether completely lacks any suggestion of streaming out, streaming out in a particular version, objects, object introduction versions, and comparisons between a requested file version and an object introduction version.

Moreover, the various elements of Appellant's claimed invention together provide operational advantages over Saether. In addition, Appellant's invention solves problems not recognized by Saether.

Thus, Appellant submits that independent claims 1, 5, and 9 are allowable over Saether.

B. Dependent Claims 2, 6, and 10 Are Patentable Over Saether

As stated above, dependent claims 2, 6, and 10 further provide that if the file version is earlier than the object introduction version, the object is represented as a proxy object (when a file is opened), and when the file is saved, the proxy object holds onto the objects data and streams out the object's data.

In rejecting these claims, the Office Action simply repeats the claim limitations and refers to Saether paragraphs [0055-0058] which provide:

> [0055] Turning to FIG. 3B from FIG. 3A, the logic steps to a block 146 where the Primary global server archives (compresses) each version change container. A third party facility may be used to implement a tape archive (TAR) command to compress each version change container. The logic moves to a block 148 where a copy of the archived version change container is encrypted and transmitted to each Secondary global server. To reduce any adverse impact on the bandwidth capacity of the network, each version change container may be broken down into relatively small units that are individually encrypted and transmitted to a Secondary global server.
> [0056] The logic moves to a block 150 where the Primary global server sends an encrypted message to each Secondary global server to unarchive the version change container. The logic steps to a block 152 where each Secondary global server unarchives the relatively small transmitted units and copies each unarchived source file to a new version in the versioned file tree repository on each Secondary global server.
> [0057] The logic flows to a block 154 where the Primary global server sends a version delivery list to each Secondary global server. In this case, the version delivery list indicates the current version, however, it should be appreciated that this list could indicate a previous version of the set of source files.
> [0058] The logic flows to a block 156 where the Primary global server and the Secondary global server build a content update container for each local content server that includes the actual file data (new source files and modified portions of previously existing source files) and indicate each source file to be deleted from the content server. The content update container is based on the two versions identified in the version delivery list. The logic advances to a block 158 where the Primary global server sends an encrypted message to each Secondary global server to copy the new and/or modified source files in the content update container to at least one sub-directory on each local content server.

These paragraphs simply provide for encrypting and transmitting an archive (from a primary server) to a secondary server (see paragraph [0055]), the primary server sending a message (to the secondary servers) to unarchive the archive, the secondary server unarchiving the archives and copying the unarchived files to a new version (see paragraph [0056]), the primary server sending a version delivery list to the secondary servers (that indicates the current version) (see paragraph [0057]), the primary and secondary servers building a content update container that includes actual file data based on the version delivery list, and the primary server sending an message (to the secondary server) to copy the new and/or archived source files from the update container to a sub-directory on local content servers (see paragraph [0058]).

However, none of these portions even remotely describe: (1) a proxy object; (2) a proxy object that holds onto object data when the object was introduced later than a requested file version; and (3) a proxy object streaming out the object's data. Instead, the cited paragraphs (and the remainder of Saether) merely describe a version change container that is transmitted between two servers with messages the follow that indicate the version of data to build and store. Such a teaching does not even remotely describe any of the claim limitations, implicitly or explicitly.

Accordingly, Appellant respectfully requests reversal of the rejections.

C. Dependent Claims 3, 7, and 11 Are Patentable Over Saether

Dependent claims 3, 7, and 11 provide the ability for superior objects to query the object to determine the appropriate version to stream out to the file. The object responds to the superior object (that actually performs the streaming out operation) to stream out in the requested file version (if the requested file version is equal to or later than the object introduction version) and in the object introduction version (if the requested file version is earlier than the object introduction version).

In rejecting these claims, the Office Action simply refers to paragraphs [0068-0077]. However, these paragraphs merely describe the hierarchy illustrated in FIG. 6. The claims provide that the superior object queries the object to determine a version to stream out in. Saether completely fails to describe any querying between objects or files in the hierarchy of file tree repository. In fact, Saether fails to describe any communication between the different files in the file tree 200. Further, the claims provide for a specific query regarding the appropriate file version to stream out to a file. No such query is taught, described, suggested, alluded to, or implied whatsoever in the cited paragraphs or the remainder of Saether.

Without teaching any communication between objects, files, not to mention the specifically claimed query, Saether cannot possibly teach or render this claim anticipated or obvious. Accordingly, Appellant respectfully requests reversal of the rejections.

VIII. CONCLUSION

In light of the above arguments, Appellant respectfully submits that the cited references do not anticipate nor render obvious the claimed invention. More specifically, Appellant's claims recite novel physical features which patentably distinguish over any and all references under 35 U.S.C. §§ 102 and 103. As a result, a decision by the Board of Patent Appeals and Interferences reversing the Examiner and directing allowance of the pending claims in the subject application is respectfully solicited.

Respectfully submitted,

GATES & COOPER LLP
Attorneys for Appellant(s)

Howard Hughes Center
6701 Center Drive West, Suite 1050
Los Angeles, California 90045
(310) 641-8797

Date: October 26, 2004

By: [illegible]
Name: Jason S. Feldmar
Reg. No.: 39,187

JSF/
G&C 30566.90-US-01

APPENDIX

1. A method for storing object data of a requested file in an object-oriented computer system, comprising:

obtaining a request to save a file in a requested file version, wherein the file contains an object;

determining if the requested file version is earlier than an object introduction version of the object;

saving the file by streaming out data representing an instance of the object to the file, in the requested file version if the requested file version is equal to or later than the object introduction version, wherein the data comprises actual methods and attributes of the object; and

saving the file by streaming out the data representing the instance of the object to the file, in the object introduction version if the requested file version is earlier than the object introduction version.

2. The method of claim 1 wherein the requested file version is earlier than the object introduction version, the method further comprising representing the object as a proxy object when a file is opened, and wherein the streaming out in the object introduction version comprises:

the proxy object holding onto the object's data; and

the proxy object streaming out the object's data.

3. The method of claim 1 further comprising:

one or more superior objects of the object querying the object to determine a version to stream out to the file;

the object responding to stream out in the requested file version if the requested file version is equal to or later than the object introduction version;

the object responding to stream out in the object introduction version if the requested file version is earlier than the object introduction version; and

the one or more superior objects of the object streaming out data in accordance with the object response.

4. (CANCELED)

5. An apparatus for storing object data of a requested file in an object-oriented computer system comprising:

an object-oriented computer system having a memory and a data storage device coupled thereto;

one or more computer programs, performed by the computer, for obtaining a request to save a file in a requested file version, wherein the file contains an object, for determining if the requested file version is earlier than an object introduction version of the object, for saving the file by streaming out data representing an instance of the object, to the file, in the requested file version if the requested file version is equal to or later than the object introduction version, wherein the data comprises actual methods and attributes of the object, and for saving the file by streaming out the data representing the instance of the object to the file, in the object introduction version if the requested file version is earlier than the object introduction version.

6. The apparatus of claim 5 wherein the requested file version is earlier than the object introduction version, further comprising one or more computer programs, performed by the computer, for representing the object as a proxy object when a file is opened, and wherein the streaming out in the object introduction version comprises:

the proxy object holding onto the object's data; and

the proxy object streaming out the object's data.

7. The apparatus of claim 5 further comprising:

means for one or more superior objects of the object to query the object to determine a version to stream out to the file;

means for the object responding to the one or more superior objects to stream out in the requested file version if the requested file version is equal to or later than the object introduction version;

means for the object responding to the one or more superior objects to stream out in the object introduction version if the requested file version is earlier than the object introduction version; and

means for the one or more superior objects of the object streaming out data in accordance with the object response.

8. (CANCELED)

9. An article of manufacture comprising a program storage medium readable by a computer and embodying one or more instructions executable by the computer to perform a method for storing object data of a requested file in an object-oriented computer system, the method comprising:

obtaining a request to save a file in a requested file version, wherein the file contains an object;

determining if the requested file version is earlier than an object introduction version of the object;

saving the file by streaming out data representing an instance of the object to the file, in the requested file version if the requested file version is equal to or later than the object introduction version, wherein the data comprises actual methods and attributes of the object; and

saving the file by streaming out the data representing the instance of the object to the file, in the object introduction version if the requested file version is earlier than the object introduction version.

10. The article of manufacture of claim 9 wherein the requested file version is earlier than the object introduction version, the method further comprising representing the object as a proxy object when a file is opened, and wherein the streaming out in the object introduction version comprises:

the proxy object holding onto the object's data; and

the proxy object streaming out the object's data.

11. The article of manufacture of claim 9 wherein the method further comprises:

one or more superior objects of the object querying the object to determine a version to stream out;

the object responding to stream out in the requested file version if the requested file version is equal to or later than the object introduction version;

the object responding to stream out in the object introduction version if the requested file version is earlier than the object introduction version; and

the one or more superior objects of the object streaming out data in accordance with the object response.

12. (CANCELED)